Kernelized Locality-Sensitive Hashing for Semi-Supervised Agglomerative Clustering

Boyi Xie
Department of Computer Science
Columbia University
New York, NY 10027
xie@cs.columbia.edu

Shuheng Zheng
Department of Industrial Engineering & Operations Research
Columbia University
New York, NY 10027
sz2228@columbia.edu

Abstract

Large scale agglomerative clustering is hindered by computational burdens. We propose a novel scheme where exact inter-instance distance calculation is replaced by the Hamming distance between Kernelized Locality-Sensitive Hashing (KLSH) hashed values. This results in a method that drastically decreases computation time. Additionally, we take advantage of certain labeled data points via distance metric learning to achieve a competitive precision and recall comparing to K-Means but in much less computation time.

1 Introduction

Our proposed research topic is to do clustering on a scalable dataset from a semi-supervised approach based on hashing methods. In particular, our goal is to explore the underlying data distribution by clustering the data points and differentiating the classes. When a small set of labeled data that come from only a subset of the classes is given, we want to find out the whole data distribution for a complete set of classes. For example, we are given a set of labels of two classes, can we separate these two classes well and at the same time discover the existence of a third class. It requires using the information from the labeled data to find a transformation metric that can split the two classes well; and after this data transformation, we can discover that there is a third class exists. Suppose there is a handwritten digit recognition task and the dataset contains digits ‘2’, ‘7’ and ‘4’. If a general agglomerative clustering is run, it might end up with 2 clusters that ‘2’ and ‘7’ in one cluster and ‘4’ in the other, due to the similarity of their shapes. However, when a small labeled set of classes ‘2’ and ‘7’ is given, we can learned a degree of granularity for similarity comparison. By using a data transformation that maximally can split ‘2’ and ‘7’ into two clusters, we are able to identify the existence of another cluster, digit ‘4’. Because agglomerative clustering suffers from its computation inefficiency, a major contribution of this paper is to introduce a machine learned hashing method - kernelized locality-sensitive hashing (KLSH) - into agglomerative clustering. This results in an efficient computation in clustering for large-scale dataset.

Our paper is structured as follows. We provide background study and related work in section 2. Section 3 presents our algorithms for distance metric learning and KLSH clustering. Section 4 describes the experiments with a discussion of the results, followed by conclusions in section 5.

2 Related Work

There has been much previous work on cluster seeding to address the limitation that iterative clustering techniques (e.g. K-Means and Expectation Maximization (EM)) are sensitive to the choice of initial starting points (seeds). The problem addressed is how to select seed points in the absence of prior knowledge. Kaufman and Rousseeuw [1] propose an elaborate mechanism: the first seed is the instance that is most central in the data; the rest of the representatives are selected by choosing instances that promise to be closer to more of the remaining instances. Pena et al. [2] empirically compare the four initialization methods for the K-Means algorithm and illustrate that the random and Kaufman initializations outperform the other two, since they make K-Means less dependent on the initial choice of seeds. In K-Means++ [3], the random starting points are chosen with specific probabilities: that is, a point p is chosen as a seed with probability proportional to p's contribution to the overall potential (defined by the sum of squared distances between each point and the closest center). By augmenting K-Means using this simple, randomized seeding technique, K-Means++ is θ(log K) competitive with the optimal clustering. Bradley and Fayyad [4] propose refining the initial seeds by taking into account the modes of the underlying distribution. This refined initial seed enables the iterative algorithm to converge to a better local minimum. Semi-supervised learning is also seen as unsupervised learning guided by constraints. Noticed that clustering is heavily dependent on distance metrics and a particular algorithm is an executor to follow the rules, [5] pointed out the desire to use a systematic way to learn distance metric for clustering from labeled data. It is based on posing metric learning as a convex optimization problem.

When the data size is growing exponentially, hashing is a technique especially good at solving large scale problems. [6] described Locality-Sensitive Hashing (LSH) method, which is an efficient algorithm for the approximate and exact nearest neighbor problem. Their goal is to preprocess a dataset of objects (e.g. images) so that later, given a new query object, one can quickly return the dataset object that is most similar to the query. The technique is of significant interest in a wide variety of areas of unsupervised learning. Hierarchical clustering tries to solve a similar problem, from another perspective. By iteratively finding nearest neighbors, it groups data into clusters. Kernelized LSH is later proposed by [7] for fast image search. It generalizes LSH to accommodate arbitrary kernel functions, making it possible to preserve the algorithm's sub-linear time similarity search guarantees for a wide class of useful similarity functions.

3 Methods

In this section, we describe our methods to solve a large scale semi-supervised learning problem by first introducing the distance learning metrics, and then our fast agglomerative clustering method based on kernelized locality-sensitive hashing (KLSH).

3.1 Distance Metric Learning

Under the circumstances that the data given to us has a few labeled points, and we know which points for sure belongs to the same or different classes. We have a similarity and a dissimilarity matrix S and D respectively. For entry $s_{i,j}$ in similarity matrix S, $s_{i,j} = 1$ if data x_i and x_j are in the same class and 0 otherwise. Similarly for dissimilarity matrix D. Based on [5], we try to learn a distance metric $||x-y||_A = \sqrt{(x-y)^T A(x-y)}$, where x, y are two data points, A is a positive semi-definite matrix of distance parameters among data points. The idea is to minimize the distance between similar points while keeping dissimilar points apart.

$$
\begin{aligned}
& min \sum_{(x_i,x_j)\in S} ||x_i - x_j||_A^2 \\
& \sum_{(x_i,x_j)\in D} ||x_i - x_j||_A \geq 1 \\
& A \succeq 0
\end{aligned}
$$

It can be solved efficiently using constrained Newton's descent on the objective function

Algorithm 1 Semi-Supervised Clustering with Hashing

Input: DataLabeled(limited), Data $x_1, ..., x_N$.
Step1: Learn distance metric *A* from labeled (limited) data
Step2: Build *A*-Distance KLSH table (Algorithm 2).
Step3:
Initialization:
• Let cluster distribution $R_0 = \{\{x_i\}, i = 1, ..., N\}$, i.e. each data point is an individual cluster C_i.
• Let proximity matrix $P_0 = P(X)$ of hash keys.
• Set $t = 1$.
repeat
• $t = t + 1$
• Find C_i, C_j such that $d(C_i, C_j) = min_{r,s=1,...,N,r\neq s} d(C_r, C_s)$.
• Merge C_i, C_j into a single cluster C_q and form $R_t = (R_{t-1} - C_i, C_j) \cup C_q$.
• Define the proximity matrix P_t from P_{t-1} by (a) deleting the two rows and columns that corresponding to the merged clusters and (b) adding a row and a column of the new cluster.
until (The remaining number of clusters is equal to a specified k; or the inconsistency coefficient exceeds a threshold.)
Step4: Retrieve actual data instances from KLSH hash table for the corresponding clusters

Algorithm 2 Build *A*-Distance KLSH Table

Input: Data $x_i, ..., x_N$, distance parameter A.
Step1: Randomly select p points from data, denoted as $x_1, ..., x_i, ..., x_p$.
Build kernel $K(i, j) = exp(-(d(x_i, x_j)^2/\sigma^2)$, for $i, j = 1, ..., p$, where $d(x, y) = \sqrt{(x - y)' A(x - y)}$.
Step2: Apply SVD to K, suppose $K = U\Sigma U^T$. $K^{-1/2} = U\Sigma^{-1/2}U^T$.
Step3: Form a p-dim vector e_s, where t dimensions are 1 while others are 0. These t dimensions are chosen randomly.
Step4: $w = K^{-1/2}e_s$. For any x, the bit is created as $h(x) = sign(\Sigma_{i=1}^{p} w_i k(x, x_i))$.

$$\sum_{(x_i,x_j)\in S} ||x_i - x_j||_A^2 - log(\sum_{(x_i,x_j)\in D} ||x_i - x_j||_A).$$

This semi-supervised part is to learn a distance metric for data transformation before the main agglomerative clustering.

3.2 Clustering with KLSH

Curse of dimensionality is a well-known problem for learning on large scale datasets. It is related to the fact that the cost of computation grows exponentially with the increase of data dimensions, or the number of data instances. This is a problem directly affects clustering approaches that based on density estimation in input space.

For instance, in K-Means or general agglomerative clustering, the cost in iteratively estimating new centroid locations and re-arranging data instances to clusters exerts a significant burden on the performance. This happens especially to dataset in high dimension, where frequently computing inter-instance distances are highly expensive.

Our proposed semi-supervised clustering algorithm using kernelized locality-sensitive hashing (KLSH) in Algorithm 1 aims to solve the large scale agglomerative clustering problem. It first learn a distance metric A from a small set of labeled data (step 1 in Algorithm 1). The second step is to build KLSH table that map the data in to hashed bits. In the rest of the procedure, an agglomerative clustering is performed. Instead of explicitly computing inter-instance distances, the clustering is done based on the KLSH-hashed data points by measuring their Hamming distance. Such kernelized locality-sensitive hashing method has a high probability of preserving neighborhoods so it's a reasonable substitute for the exact inter-instance distances.

Table 1: Experiment results that compare four methods: (1) K-Means, (2) K-Means with Distance-metric Learning, (3) Agglomerative clustering using KLSH and (4) Agglomerative clustering using KLSH with Distance-metric Learning. Precision, recall and computation time is reported. They all run on data underlying 10 classes and the hash code is 32-bit.

# Inst.	K-Means			K-Means w/ DL			Aggl. KLSH			Aggl. KLSH w/ DL		
	Pre	Rec	Time	Pre	Rec	Time	Pre	Rec	Time	Pre	Rec	Time
5000	.590	.564	13.246	.537	.496	15.647	.573	.305	2.155	.631	.272	2.466
10000	.568	.540	46.398	.580	.556	33.736	.520	.336	7.255	.613	.250	5.246
15000	.574	.530	69.252	.556	.539	186.469	.584	.180	13.843	.610	.156	8.077
20000	.589	.563	79.499	.455	.448	112.178	.609	.355	3.052	.617	.292	18.070
30000	.523	.503	164.853	.552	.541	139.773	.624	.235	58.646	.548	.306	23.136
50000	.560	.531	339.599	.565	.530	333.313	.579	.230	126.280	.590	.252	122.558

4 Experiments

Our experiment is based on the MNIST dataset of handwritten digits. We evaluate our KLSH agglomerative clustering algorithm via a comparison to K-Means.

4.1 Datasets

We obtained handwritten digits from the MNIST data repository. There are 10 classes of rasterized images (corresponding to digits from '0' to '9'). We used up to 50,000 data points for experiments.

4.2 Experiment Setup

We ran experiments using both K-Means and KLSH agglomerative clustering with and without distance metric learning. Hash string length, the number of classes, and the number of data points are varied one at a time. We report precision, recall and the computation time. All the experiments were done on a machine with 8-core Intel processors of 2.8 GHz and 8 GB of RAM.

4.3 Results and Analysis

Tables 1-3 summarize our results, and the followings are several trends to notice.

First in Table 1, we observed that KLSH agglomerative clustering can achieve the same level of precision for a fraction of the computational cost. The downside is that recall is caused by the factor of 2. The decrease in recall is caused by the fact that KLSH cannot recover all of the points in the nearest neighborhood. The addition of distance metric learning has noticeable benefits on performance for KLSH Agglomerative Clustering.

In Table 2, we analyzed the effect of an increase in the number of classes (while fixing the number of data points) on precision, recall, and computation time. Precision remains constant while recall decreases. The computational costs remains relatively independent of the number of clusters.

In Table 3, we analyzed the effect of hash string length on clustering validity. Increasing the length of hash string increases both the precision and recall.

It is also able to adjust the tradeoff between efficiency and effectiveness. Notice that even if we use $m = 32$ bit binary hash code, there are still 2^{32} possible outcomes. If the hashing split data well, the number of entries of the table will still be very large. It increases the accuracy of clustering results but meanwhile leads to a higher computation cost during agglomerative clustering.

According to the results, clustering with KLSH has superior performance when the dataset is large and the number of real clusters is small. Comparing to K-Means, it has large promising improvement on speed. When true cluster number is not large, it achieves high performance on both speed and accuracy. Especially in a lower level of the linkage tree, clustering with bias (distance metric learning) can immediately correctly cluster similar data instances.

Table 2: Compare the performance of various number of underlying classes for agglomerative clustering using KLSH with distance metric learning. In this case, data size is 20,000 and the hash code is 32-bit. With an increase in the number of classes, the precision remains constant while recall decreases

# CLASSES	PRE	REC	TIME
4	.714	.465	16.527
5	.629	.355	19.226
6	.702	.507	18.747
7	.611	.317	21.675
8	.654	.354	21.540
9	.603	.284	24.024
10	.617	.292	18.070

Table 3: Compare the performance of various number of hash code bits for agglomerative clustering using KLSH with distance metric learning. In this case, data size is 20,000 with underlying 10 classes. Increasing the length of hash string increases both the precision and recall.

# BITS	PRE	REC	TIME
8	.402	.245	0.043
16	.599	.111	1.000
32	.617	.292	18.070
64	.635	.380	92.452

5 Conclusions

General hierarchical clustering methods cannot scale well on large dataset due to the exponentially growing number of calculations on inter-instance distances. Kernelized locality-sensitive hashing (KLSH) provides a high probability of preserving neighborhoods and it's a reasonable substitute for the exact inter-instance distances. Our proposed KLSH agglomerative clustering alleviates the problem by calculating a reduced-sized Hamming distance and achieves efficient clustering computation. The incorporation of distance metric learning marginally improves the precision and recall.

References

[1] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley, 1990.

[2] José Manuel Peña, José Antonio Lozano, and Pedro Larrañaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters*, 20(10):1027–1040, 1999.

[3] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007.

[4] Jude W. Shavlik, editor. *Refining Initial Points for K-Means Clustering*. Morgan Kaufmann, 1998.

[5] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, pages 505–512. MIT Press, 2002.

[6] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, January 2008.

[7] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.